Integrated cloud based computational services

Lapin, Andrei ; Schiller, Eryk

Abstract: This paper describes a general architecture of a cloud integrated solution for implementing tailored use-cases of various research communities. A successful implementation of a hydrogeological modeler deployed in the cloud proves the potential of the architecture. The knowledge and experience gathered throughout the hydrogeological use-case allow us to specify yet another similar scenario - a future NS3 modeler in the cloud for researchers of the networking domain.

Integrated cloud based computational services

Andrei Lapin[1], Eryk Schiller[1]

[1]University of Neuchâtel, Switzerland

Abstract: This paper describes a general architecture of a cloud integrated solution for implementing tailored use-cases of various research communities. A successful implementation of a hydrogeological modeler deployed in the cloud proves the potential of the architecture. The knowledge and experience gathered throughout the hydrogeological use-case allow us to specify yet another similar scenario–a future NS3 modeler in the cloud for researchers of the networking domain.

1 Introduction

Currently, cloud computing is a growing business with many players. Among the most successful ones there are Salesforce, Amazon, Akamai, and Google. A cloud infrastructure provides users with great advantages:

- Reduction of CAPital EXpenditures (CAPEX) since there is no need to maintain a hardware infrastructure,
- Full control of required and purchased resources,
- Improved accessibility, because users are always provided with access from anywhere in the world.

In academia, there is an ever growing number of scientific domains that require large computing power spent on various research projects. Often, scientific problems are similar in terms of execution mechanisms, i.e., a research application has to be executed many times with various parameters. In this work, we provide an infrastructure which distributes independent (easily parallelizable) repetitive scientific processes within a cloud-based computation environment for the benefit of scientific communities. A generic architecture allowing for the implementation of various scientific use-cases is provided. It benefits out of constantly growing cloud-based computational resources.

This work has the following organization. Section 2 provides a general overview of the architecture. Section 3 tackles the deployment of a hydrogeological use-case in the cloud. In Section 4, we describe the architecture of the future cloud-based NS3 deployment. Finally, we conclude in Section 5.

2 Generic architecture

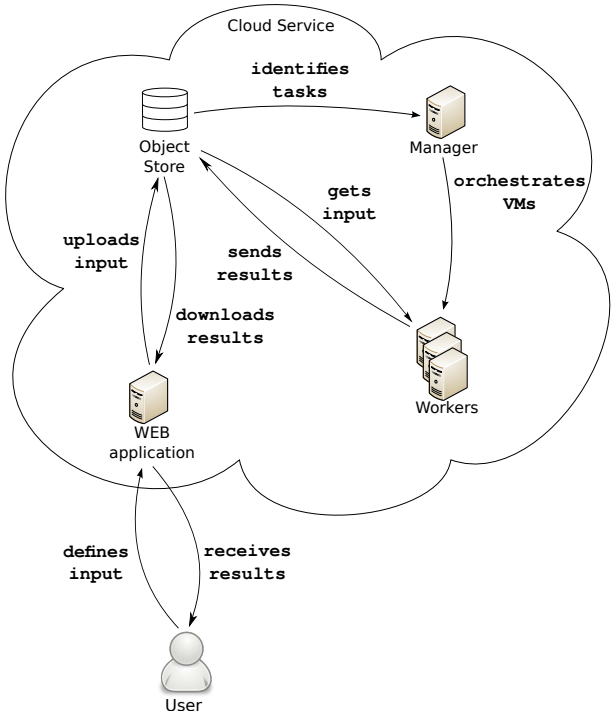

Fig. 1: General architecture

Figure 1 illustrates a general architecture of our system. This general architecture allows us to move computationally expensive and time consuming tasks of scientific processes into clouds. We deployed a leader machine in the cloud, i.e., a manager, responsible for orchestrating other elements. Each scientific process occupies a single virtual machine (VM). The Manager spawns working VMs for newly scheduled processes. There are also monitoring functions implemented between VMs and the Manager. The end-user has a top-level view of the system, since s/he only communicates with the Manager through the web interface.

The Manager is built upon the GC3Pie [2]–a Python framework which was implemented by the Grid Computing Competence Center (GC3) of the University of Zurich and allows us to quickly develop a robust managing solution. Working machines are provided with a specialized Linux-based image (Ubuntu 13.10) with required software and configuration integrated [4]. As general cloud storage, we use Amazon's S3 compatible ObjectStore in which we collect input, output as well as internal intermediate states of other system components. The web interface is implemented in Django–a Python web framework to keep the implementation homogeneous in terms of programming languages and to benefit out of already implemented modules (e.g., the S3 Interface) in various system components (i.e., Manager, web interface). In Section 3, we test this architecture provided in the case of a hydrogeological use-case.

3 HydroGeoSphere application in the cloud environment

Hydro-geological research often requires complex environmental simulations to predict the behavior of a real environmental system in the future. The HydroGeoSphere [6] (HGS) is one of the most extensively used applications for simulating ground water and surface water profiles for a certain region. It executes physics-based models between these two water profiles [3, 1]. A single model recalibration of the Emmental region[1] takes a long time on an ordinary desktop machine, i.e., approximately 3–4 hours. Moreover, it is usually required to run multiple models (for example 100) to achieve predictions of good quality (e.g., in the case of the Ensemble Kalman Filter approach). Hydrogeologists require recalibrating all the models in near real-time (up to a few hours) to provide a reliable and up-to-date forecast. The near real-time model recalibration condition requires us to use huge computation power. Fortunately, in the Ensemble Kalman Filter[2], these simulations are completely independent and therefore the whole process is highly suited for parallelization. Therefore, our architecture described in Section 2 perfectly matches the requirements and allows to simultaneously execute a large number of HGS in the cloud thus providing the model recalibration in near real-time [5].

[1] The Emmental is a pre-alpine river valley in central Switzerland.
[2] Ensemble Kalman Filter is a Monte Carlo simulation.

3.1 Architecture

Figure 2 presents an architectural view of the system. Since the HydroGeoSphere application requires real-time data from environmental sensors to run simulations, our architecture consists of two main building blocks 1) a Wireless Mesh Network for environmental monitoring and 2) a Cloud Computational Service for environmental modeling. The monitoring infrastructure consists of a wireless mesh network in a remote region and environmental sensors attached to wireless nodes. The network is constantly connected to the Internet through a gateway. We also adapted Zabbix [7]– a network monitoring infrastructure to monitor states of our environmental sensors. This allowed us to gather environmental data from our mesh network in real-time and push it to the database. Later on, the cloud computation service can pull the data and use it in simulations.

The HGS is an application which computes hydro-geological models; it does not need any interactions with the user after the command is executed. Therefore, we decided to simplify and adapt our generic architecture to these specific properties. The main simplification is that the user does not need to interact with working machines, because s/he only schedules new computing and receive results of a computation. We can thus isolate and automate the VMs launching process. In our system, the Cloud Task Manager (CTM) handles this process. It periodically checks the input directory of the ObjectStore for newly available input installed by the user through the web interface. If the new input is found, the CTM deploys it on a vacant or newly created VM and launches new computations. The initialization and monitoring processes bring an additional delay to the overall turnaround time, but this delay is insignificant in comparison to the model runtime of 3–4 hours. The CTM installs the outcome of the computation in the ObjectStore; the user can download the outcome by listing the output bucket of the ObjectStore through the web interface.

3.2 Comparison

The cloud-based execution brings significant advantages in terms of computation time. Figure 3 shows a time comparison between sequential execution and cloud-based simultaneous execution. We denote the number of computed models N along the x-axis. The y-axis is the required application execution time, where t is an average model execution time and d is a new model identification

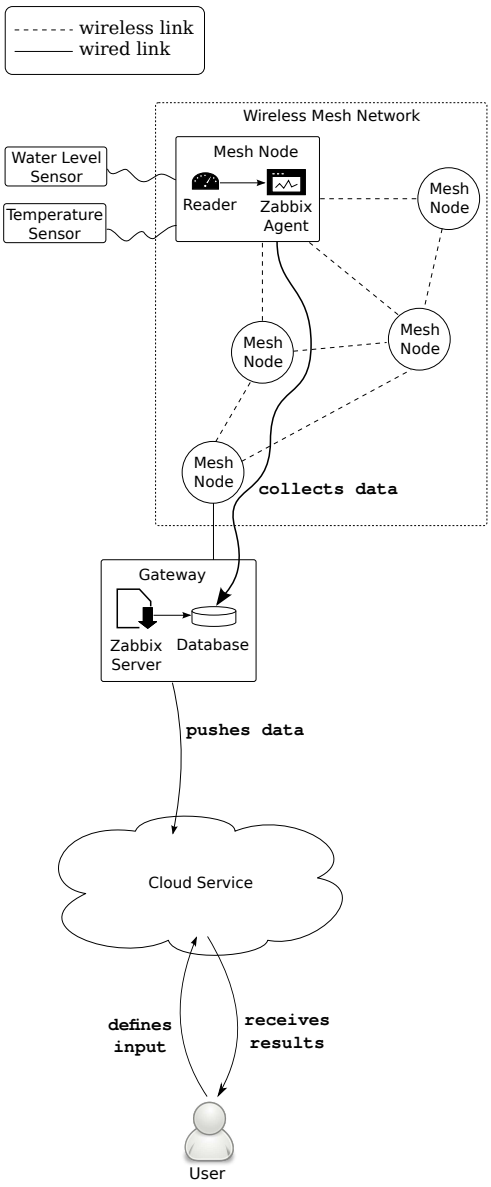

Fig. 2: Environmental modelling system architecture

time in the cloud environment. We suppose that t gets much larger values than d, e.g. t equals 3–4 hours, d equals 1–5 minutes. In the case of sequential execution, the execution time grows linearly with respect to the number of computed models. On the other hand, in the ideal case, for the cloud-based execution, the execution time should stay at the $t + d$ level. We should, however, also consider possible cloud-specific restrictions, e.g. the availability of virtual resources (for example to run a subsequent simulation, we have to wait until all the necessary resources such as VMs become free, hence the execution time increases as a step function), the ObjectStore performance and the load of the cloud (performance drops "continuously" through the increase of the completion time). In

any case, taking into account all these restrictions, the cloud-based execution time should get much smaller values than the sequential execution time on an ordinary desktop machine.

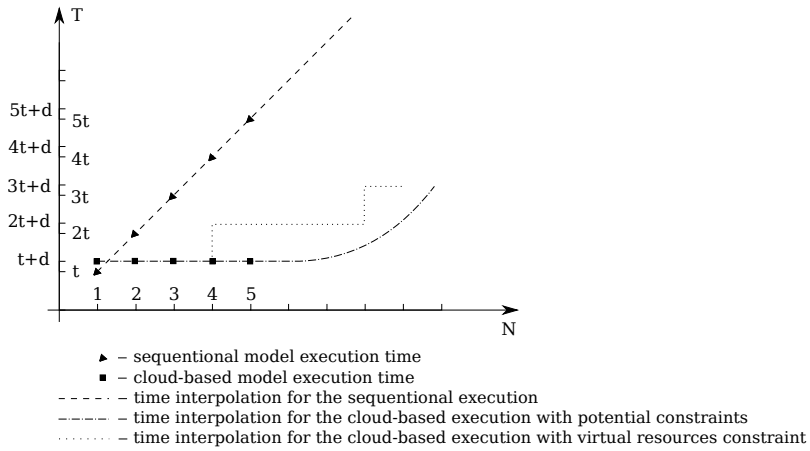

Fig. 3: Execution time comparison

4 NS3 simulator in the cloud environment

Previously described requirements of the hydro-geological domain also perfectly match the network simulations in the cloud architecture. In this section, we describe "easily parallelizable repetitive processes" of the NS3 simulator which can go to the cloud environment.

In the networking domain, simulating the behavior of new protocols in various topologies and network conditions is an important aspect. Before implementing a new protocol in the real environment, network researchers study its behavior in network simulators. NS3 is a discrete event-based network simulator targeting primarily research and educational use. An average usage scenario is the following. The user prepares the simulation and determines necessary modules of the simulator. If necessary s/he modifies the required module basis or extends it with his/her own modules developed. Then, s/he writes a simulation program, which describes the whole simulation process, and also defines the input and output. The next step is to launch the simulator with appropriate parameters. Finally, when the simulation completes, the user downloads

the results and studies their content. The described usage scenario allows us to successfully move the simulation process to the cloud environment, because the user deals with independent simulations which can run on VMs in the cloud.

4.1 Advantages/inconveniences of the cloud environment

The cloud environment should bring considerable benefits while the user is provided with significant advantages compared to the current sequential simulation process:

- Resource control; the user can appropriately set up VMs for each particular simulation with required parameters such as CPU, RAM, etc.,
- The user can edit simulation scripts / source code and launch new simulations immediately after launching the previous one without a fear of a machine overload,
- The user can run a large number of parallel simulations on many simultaneous VMs,
- The result gathering process is centralized and the user always has easy access to outcome and corresponding simulation scripts,
- Controlling the simulation process from any device with the Internet connection, also online editing of scripts/source files shall be possible,
- Easy, semi-automatic process of setting-up the working environment; knowledge of the cloud infrastructure is not necessary,
- Simple migration of the working environment among VM-based clouds, batch-queuing clusters, computational Grids, Linux/UNIX hosts in which a user is provided with SSH-based access.

Possible inconveniences:

- The user cannot use a working front-end of her/his choice, such as Eclipse to edit simulation scripts, but s/he has to use the online environment provided. We can overcome this inconvenience by providing an upload mechanism of locally written scripts,
- Availability; Internet connectivity is required to set up a simulation. When the user launches a simulation, the remaining process is, however, fully automatic up to saving results to persistent storage.

- Potential costs; Usually cloud providers charge users for access to the infrastructure. Our solution shall be compatible with every VM based cloud, therefore the user will not be bounded to a specific provider.

4.2 Architecture specific details

This use-case differs from the previous one, so it requires additional modifications of the general architecture. The output-providing mechanism must be carefully designed, because the NS3 simulator offers various output information which depends on the user's requirements, e.g. text output, graphics, pcap information, compilation results. Thus, the user has to have access to all output information and be able to get it on demand.

We designed a three-level structure shown in Figure 4 to store the input information for the simulations within the ObjectStore. ObjectStore is the first level and the root element of the structure. At this point, we distinguish different users of the service; each user has his/her own directory to store all implemented simulations. The set of user directories is the second level of the structure; the files provided by two different users are not to be mixed. The third and the last level of the structure is a set of simulation archives inside the user's input directory. Each particular simulation completely describes the required simulation and contains all the changes provided by the user from the default NS3 simulator, e.g., configuration files, simulation scripts, changes in the source code of the main simulator tree. This input structure fully guarantees a clean and transparent input storage mechanism and a minimal number of transactions to the ObjectStore when requesting input for a simulation.

For output, we designed a four levels structure, presented in Figure 5. The first two levels are equivalent to the input structure, i.e., the root element and user directories. The third level is a set of simulation directories; each simulation output directory should have an identical name with the corresponding input archive thus guaranteeing a clear relation between these two. One simulation might have different types of output information, therefore, at the fourth level, we provide a list of available outputs for the simulation and corresponding output archives for each particular output type. This structure also guarantees a clean and transparent output storing mechanism and minimal number of transactions to the ObjectStore for obtaining output upon the user's demand.

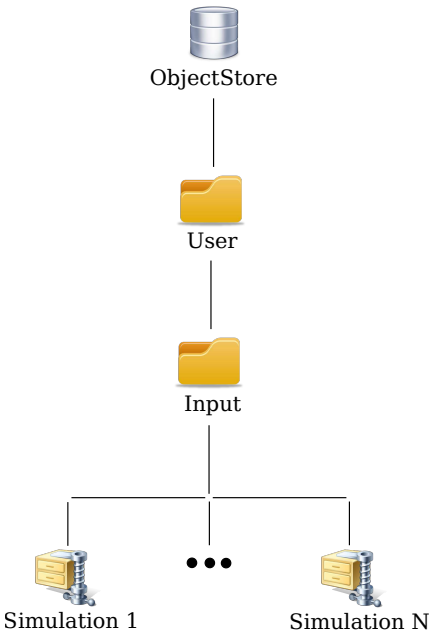

Fig. 4: Structure of the input directory in the ObjectStore

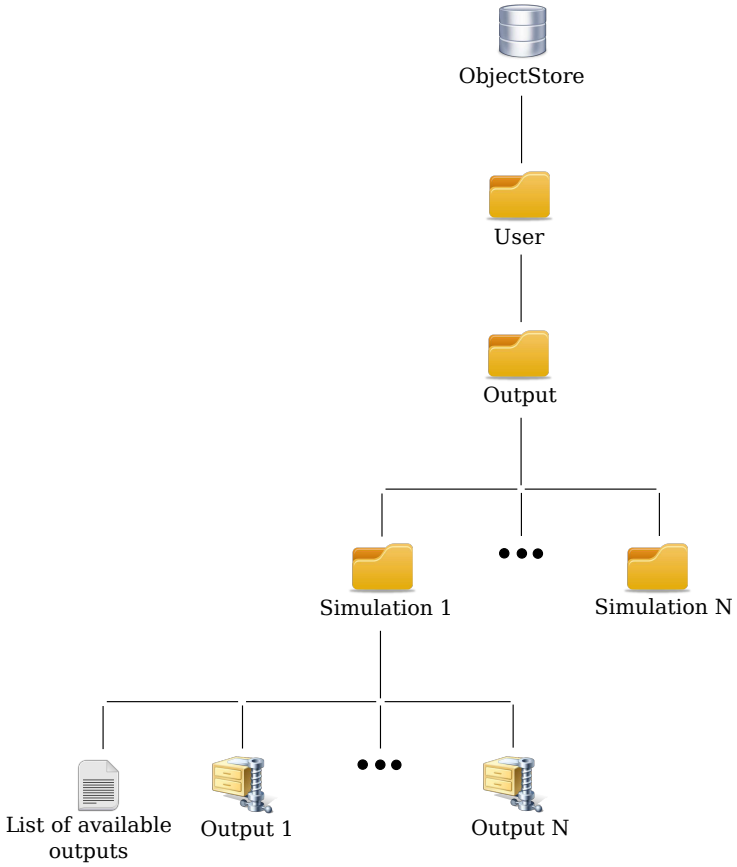

Fig. 5: Structure of the output directory in the ObjectStore

4.3 Challenges

An additional challenge in the design arises due to the fact that we can distinguish two different usage scenarios 1) the user changes the source code of the simulator, therefore the appropriate modules of the simulator have to be recompiled 2) the user only changes the corresponding simulation script, hence only the script has to be recompiled. The first scenario might take a long time, depending on the user's modifications, because the source code of the simulator contains thousands of code lines with complex dependencies. On the other hand, the second scenario might be more dynamic in terms of execution time for simple simulations, thus launching and terminating VMs processes would be extremely time-consuming, however, still required for long time-consuming models. An optional mechanism for launching consecutive simulations only on a single VM must be designed. We foresee a pool with a configurable number of launched VMs to speed up the execution process time. In this case, when the user launches a new simulation, the NS3 simulator can immediately begin executing it.

Another challenge that we face here is VM resource-allocation. We need to design an algorithm which is able to predict the necessary amount of virtual resources for user's simulations. Predictions should be based on statistics of previously-launched simulations, as well as source code modifications made by the user. In such a case, we can optimize resources, which leads to both minimal simulation execution time and fair resource distribution among VMs. In some cases, we can also depend on the user meta-data provided indicating whether the user runs a long simulation (therefore s/he does not care about the additional overhead of launching a new machine) or s/he just launches a small simulation (in such a case, the user does not care much about the computing performance, but she wants to quickly get results such as consistency of the simulation; possible allocation on an already running machine might be taken into consideration in this case).

4.4 System processes

Figure 6 shows the system components interaction. The user interacts with the web interface through which s/he can perform all the necessary modifications to the simulations. Then the Job Controller assembles a new task and sends it

to the Cloud Task Manager. When the CTM receives a new task, it queues it for future processing by VMs.

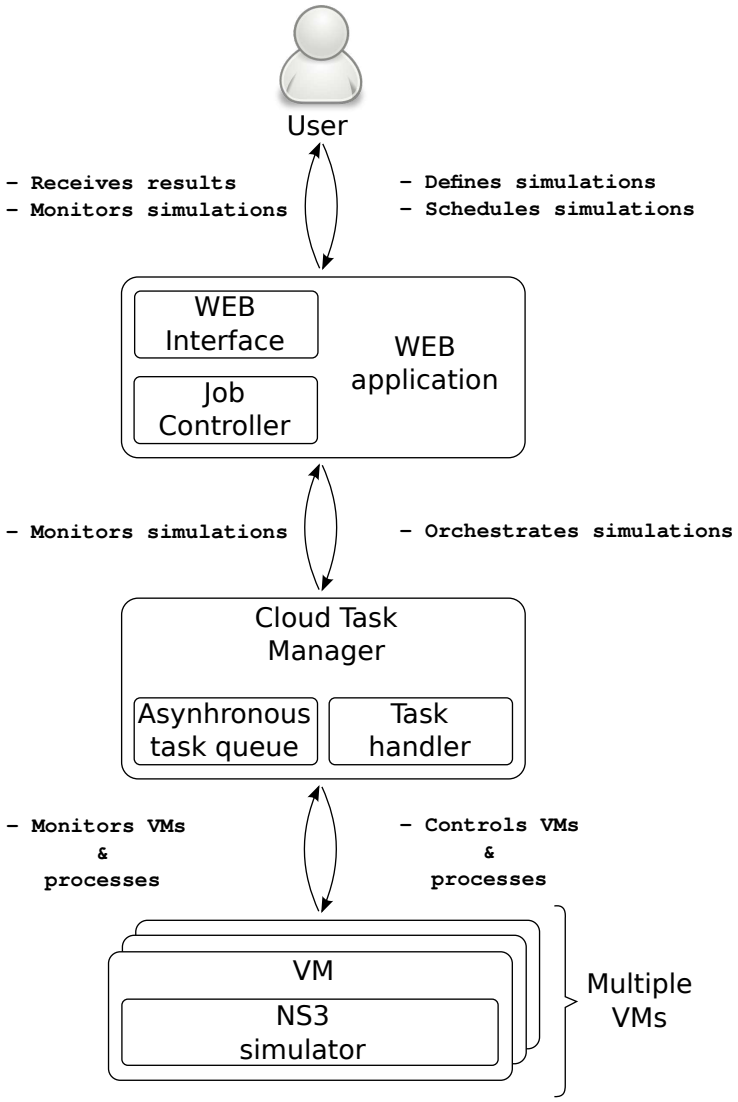

Fig. 6: System processes diagram

We defined general functionality and a primary architecture of the cloud service. The final design requires careful and detailed studies on different aspects of the performance objective function. For this reason, we need to collect user-usage statistics for the NS3 simulator and use this information for the final system scheduling algorithm.

5 Conclusion

Many research domains require large computation power to perform scientific experiments, therefore, a modern and effective computational approach is required. In this paper, we have introduced a general cloud-based infrastructure, which can easily adapt to many scientific domains for the benefit of various scientific communities. We have also described our successful migration of a hydro-geological application to the cloud environment and our ongoing efforts to adapt the NS3 network simulator to the cloud.

Bibliography

[1] Philip Brunner and Craig T. Simmons. "HydroGeoSphere: A Fully Integrated, Physically Based Hydrological Model". In: *Ground Water* 50.2 (2012), pp. 170–176. ISSN: 1745-6584. DOI: 10.1111/j.1745-6584.2011.00882.x. URL: http://dx.doi.org/10.1111/j.1745-6584.2011.00882.x.

[2] University of Zurich Grid Computing Competence Centre. *gc3pie—Python libraries and tools for running applications on diverse Grids and clusters*. https://code.google.com/p/gc3pie/. 2010.

[3] Peter Kropf et al. "Wireless Mesh Networks and Cloud Computing for Real Time Environmental Simulations". In: *Recent Advances in Information and Communication Technology*. Ed. by Sirapat Boonkrong, Herwig Unger, and Phayung Meesad. Vol. 265. Advances in Intelligent Systems and Computing. Springer International Publishing, 2014, pp. 1–11. ISBN: 978-3-319-06537-3. DOI: 10.1007/978-3-319-06538-0_1.

[4] Peter Kunszt et al. "Towards a Swiss National Research Infrastructure". English. In: *Euro-Par 2013: Parallel Processing Workshops*. Ed. by Dieter an Mey et al. Vol. 8374. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 157–166. ISBN: 978-3-642-54419-4. DOI: 10.1007/978-3-642-54420-0_16. URL: http://dx.doi.org/10.1007/978-3-642-54420-0_16.

[5] Andrei Lapin et al. "Real-time Environmental Monitoring for Cloud-based Hydrogeological Modeling with HydroGeoSphere". In: *The First International Workshop on HPC Applications associated with the High Performance Computing and Communications Conference, IEEE HPCC'14*. 2014.

[6] R. Therrien et al. *HydroGeoSphere: A Three-dimensional Numerical Model Describing Fully-integrated Subsurface and Surface Flow and Solute Transport*. Tech. rep. Waterloo, Ontario, Canada, 2007.

[7] Alexei Vladishev. *Zabbix home page—an Enterprise-Class Open Source Distributed Monitoring Solution*. http://www.zabbix.com/. 2001.